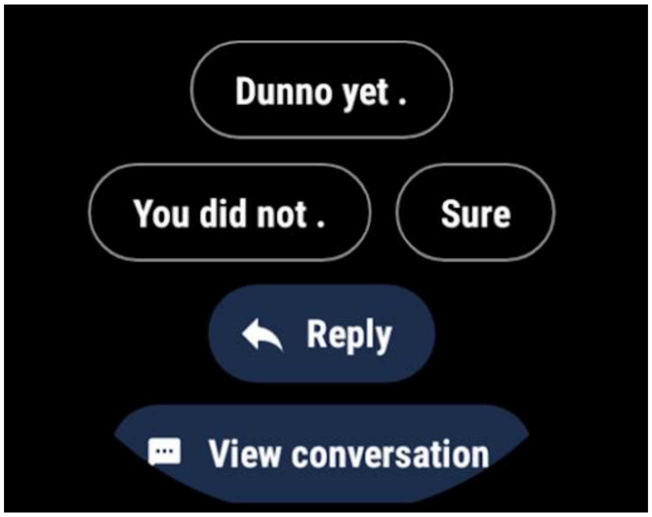


AUTOMATED RESPONSE SUGGESTION

Using Natural Language Processing

Final Project Report

Ashish Reddy Podduturi
Net ID: ap1822

Ushasee Das
Net ID: ud29

Advanced Analytics Practicum

Introduction

Infragistics, a software solutions company, plans to incorporate the Smart Reply functionality in its chat platform to enhance the user experience. As a part of this project, we will partner with Infragistics and investigate the problem of predictive response generation on a chatting platform and propose a model for the Smart Reply' feature. Organizations typically use their chat platforms to communicate, collaborate and share information promptly and more effectively. The motivation came for this project when we questioned how we can assist the user in delivering best fit quality responses with just one click. This type of suggestive replies are especially useful in mobile devices. Although we will focus our research on publicly available datasets, Infragistics can benefit from our work on this predictive analytics and can train and build similar models on its user data.

Background

Infragistics is a global software company, founded in 1989, that publishes user interface (UI) development tools and components for a range of developer applications, across all platforms. The company is also a provider of developer support, testing tools, and UI and User Experience (UX) training and consulting services.

Their products enable developers to create UIs that are the foundation for developing applications with data visualization in line of business applications for platforms that include —Windows Forms, Windows Presentation Foundation (WPF), ASP.NET and Silverlight as well as jQuery/HTML5, and mobile controls for the Windows Phone, iOS (iPhone and iPad) and Android.

Infragistics branched into the application prototyping industry with Indigo Studio, which is aimed at injecting simple design principles into the software development process. Additionally, in 2012 Infragistics acquired mobile app development company, Southlabs and continues to develop their Enterprise Mobility products, SharePlus (a mobile sharepoint client) and ReportPlus (a mobile dash boarding application).

The company, based in Cranbury, NJ, was established in November 2000 when ProtoView Development Corporation and Sheridan Software Systems, Inc. merged. Besides its U.S. headquarters in New Jersey, Infragistics has offices in eight other countries including the UK, Germany, Australia, France, Japan, India, Bulgaria and Uruguay.[1]

All the model building, and visualization has been done on Python and using python-based technologies like Pandas, Matplotlib, Pytorch and TensorFlow. The final model is LSTM (Long Short-term Memory Neural Network) model with clustering methods to tie in semantically similar sentences. The LSTM will omit probabilities for target sentences depending on the input sentences. The learning rate and regularization rates will be determined by cross validation of the model. First, we create response sets, which are semantically close sentences that could serve as responses for wide variety of User inputs. These are generated by converting target texts into vectors/embeddings and clustering them in N dimensional space with their own labels. Once the clustering is done, we can train the model to map input

[1] https://en.everybodywiki.com/Infragistics

texts to the clustering labels of target texts. For clustering together all similar sentences, we use the DBSCAN (Density-Based Spatial Clustering of Applications with Noise)i algorithm. Now a LSTM model takes input text embeddings as input with two dense layers with sigmoid activation function and a dropout layer of 20%. The dense layer outputs the probability of each target text clustering label. We take clusters with top five probability scores and picking any random response from the cluster should provide us five possible options which we can cut down to three from probability score.

Literature Review

While reviewing literature research on related topics, we mostly noted works on chatterbots or chat bots which engage with users using a predefined standard template of guidelines. What we are trying to build here is an automated response generating model which would suggest Smart Replies for a messaging system. The two most related publications that we found are briefed below.

[2]This research work done by the Google team focuses on proposing a novel end to end approach for generating automatic Smart Replies for Gmail Inbox. The paper describes an approach where they use a long short-term memory (LSTM) neural network model to preprocess any incoming emails and predict sequences of response. Since this model is computationally expensive, only the best fit responses are selected to make the model more scalable. Next, using a semi supervised graph learning model, a small but highly effective response set is generated to provide the best quality user experience. They generated diverse semantic intent clusters using two strategies: omitting redundant responses and enforcing negative or positive responses. Finally, a feedforward neural network model is built to decide if at all any suggestive response is required. This triggering model is applied to every incoming emails to rule out the ones which do not require Smart Replies such as promotional emails, no-reply emails, auto-generated emails or emails with sensitive contents. To evaluate triggering model performance, standard metrics – Precision, Recall and area under ROC Curve were used. The authors observed that it was okay to over-trigger the system as the cost for response suggestion triggers seemed quite low. LSTM scoring model was evaluated based on Perplexity, Mean Reciprocal Rank and Precision@K. The authors concluded the model was a success considering the fact that 10% of mobile replies in Gmail Inbox are sent using the Smart Reply system.

[3]Efficient Natural Language Response Suggestion for Smart Reply - This paper proposes a natural language processing Smart Reply model used by Google which is computationally more efficient compared to LSTM model described in the publication [1]. The proposed model delivers the same quality optimized response suggestion but at the cost of a minimal fraction of computational requirements and latency. The approach shared is a "feed-forward neural networks model using n-gram embedding features to encode messages into vectors which are optimized to give message-response pairs a high dot-product value. The factorized dot-product scoring model finds response suggestions with the highest score where scoring function is the dot-product. They proposed an efficient nearest-neighbor search of the hierarchical embedding of the responses to find the best response suggestions" [2]. Finally,

[2] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann and A. Tomkins, "Smart Reply: Automated Response Suggestion for Email," 2016. [Online]. Available: https://www.kdd.org/kdd2016/papers/files/Paper_1069.pdf.
[3] M. HENDERSON, R. AL-RFOU, B. STROPE and Y.-H. SUNG, "Efficient Natural Language Response Suggestion," 2017. [Online]. Available: https://arxiv.org/pdf/1705.00652.pdf.

they enforced diversity for target responses using a semantic intent clustering algorithm, same as defined in the first paper.

[4]OCC: A Smart Reply System for Efficient In-App Communications is the method used by Uber. OCC means one-click-chat feature which aims to provide their driver partners to quickly reply to their riders by providing options of replies using Machine Learning and Natural language Processing Techniques. They divide the problem into two parts (1) Intent detection and (2) Reply Retrieval. The intent detection system detects the intent of incoming messages to the driver partner and reply to the retrieval part retrieves the relevant responses to the incoming message based on intent detected. For training their model, they used their in-app encrypted and anonymized data. For intent detection they converted the messages into embedding using Doc2Vec method. This model helps in understanding the semantic meaning of the words. They used Nearest Neighbor Classification to intent detection. So, any new message which is converted to embedding is mapped on the classifier and depending on its distance from each intent centroids, new messages intent is estimated. In the reply retrieval they are leveraging historical message pairs to detect most frequent replies for each intent class and they use character level Convolutional Neural Network (Char-CNN) along with other models but Char-CNN performed with high prediction frequency of 77.2%.

[5]Data-Driven Response Generation in Social Media – This paper focuses on a similar problem of automatic response generation for social media platform Twitter. Here the proposed machine learning model is built on phrase-based Statistical Machine Translation (SMT) where the system translates and generates responses for Twitter status posts. BLEU is the automatic evaluation metric used for measuring the model performance.

[6]Preprocessing Techniques for Text Mining - An Overview – The research helped us to learn about text mining, application of text mining and its pre-processing techniques. The research gives an overall understanding of extraction with tokenization, Stop words Removal, Stemming algorithms and TF/IDF techniques.

[7]Smart Reply Generation for SMS Using Natural Language Processing – This paper uses the Ubuntu corpus dataset for their model building. They have three approaches for models which are Term Frequency - Inverse Document Frequency (TF-IDF), Recurrent Neural networks (RNN), and Long Short-Term Memory (LSTM). They've identified that LSTM is more suitable for this task after testing their performances. They've finally built a sequential LSTM with dense sigmoid activation layers.

Data Overview

We wanted to collect chat datasets that are openly available. Even with in the chat datasets that are available, the datasets are appropriate for this business use cases are general, and professional conversations and chat messages between humans. We observed multiple chatbot

[4] Y. Weng, H. Zheng, F. Bell and G. Tur, "OCC: A Smart Reply System for E€icient In-App Communications," 2019. [Online]. Available: https://arxiv.org/pdf/1907.08167.pdf.
[5] S. Vijayarani, J. Ilamathi and Nithya, "Preprocessing Techniques for Text Mining - An Overview," [Online]. Available: https://www.researchgate.net/profile/Vijayarani-Mohan/publication/339529230_Preprocessing_Techniques_for_Text_Mining_-_An_Overview/links/5e57a0f7299bf1bdb83e7505/Preprocessing-Techniques-for-Text-Mining-An-Overview.pdf
[6] M. Kaumada, S. Sumanasekara, N. Jayasekara and S. Wimaladharma, "Smart Reply Generation for SMS Using Natural Language Processing," 2020. [Online]. Available: Smart Reply Generation for SMS Using Natural Language Processing.
[7] R. Pujari, "Training Your Own Message Suggestions Model Using Deep Learning," 16 April 2021. [Online]. Available: https://towardsdatascience.com/training-your-own-message-suggestions-model-using-deep-learning-3609c0057ba8.

datasets for Natural Language Processing purposes/model building including movie subtitles dataset, Ubuntu technical support chat dataset but these datasets have a bias towards specific topics like Ubuntu has technical jargon about Linux commands and movie subtitles have slang that probably would not be used in professional setting. We also looked Chatbot dataset which was retrieved from Kaggle account of Arnav Sharma AS which was altered by him to remove any biases from its previous version and is available in CSV format. It is Chatbot dataset originally received from amazon with 8000 conversations and 184000 messages from those conversations. But this dataset was considered inappropriate for our use case with its conversations being about general knowledge. So, we chose to use three datasets:

1. Amazon Question and Answer dataset
2. Twitter Support chat dataset
3. Custom dataset prepared by us

Data Description and Cleaning

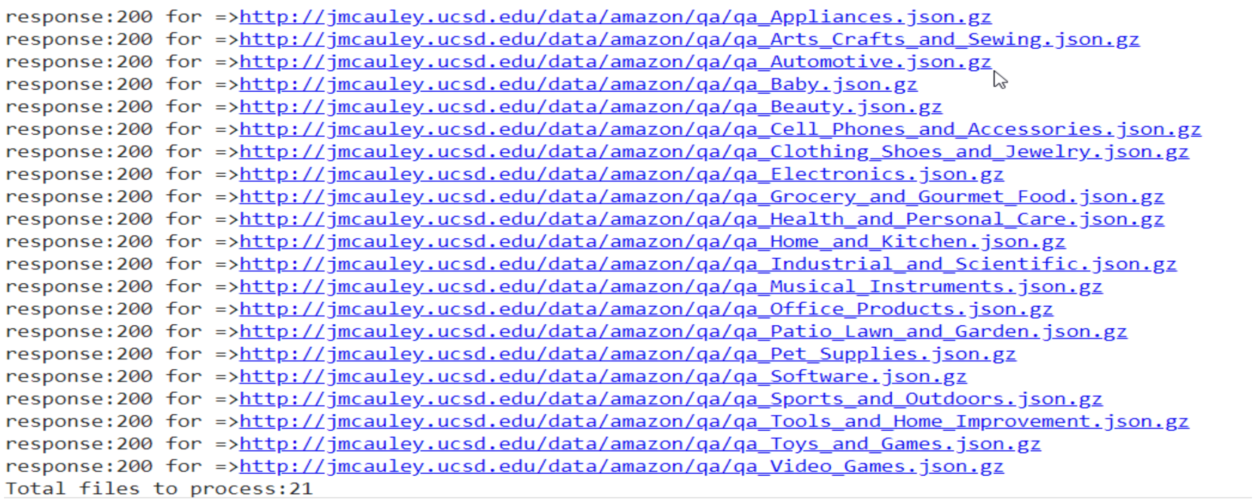
Amazon QA Dataset

The link to Amazon QA datasets can be found here. This dataset contains Question and Answer data from Amazon, totaling around 1.4 million answered questions from 1996 to 2014. All these datasets are available in zipped json format in the following 21 categories.

Categories	Questions
Appliances	9,011
Arts Crafts and Sewing	21,262
Automotive	89,923
Baby	28,933
Beauty	42,422
Cell Phones and Accessories	85,865
Clothing Shoes and Jewelry	22,068
Electronics	314,263
Grocery and Gourmet Food	19,538
Health and Personal Care	80,496
Home and Kitchen	184,439
Industrial and Scientific	12,136
Musical Instruments	23,322
Office Products	43,608

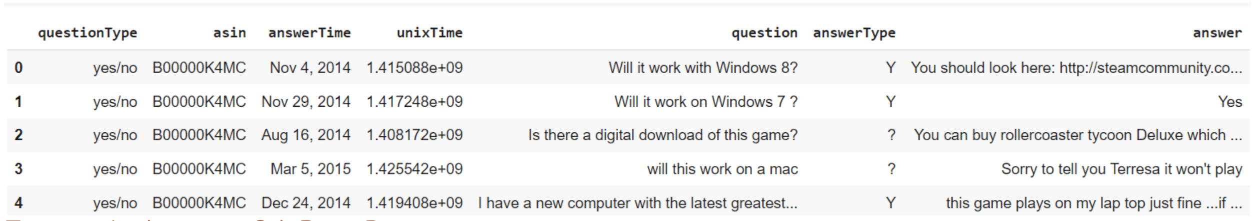
Patio Lawn and Garden	59,595
Pet Supplies	36,607
Software	10,636
Sports and Outdoors	146,891
Tools and Home Improvement	101,088
Toys and Games	51,486
Video Games	13,307

All this raw QA data was downloaded, compiled and stored in a single data frame.



```
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Appliances.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Arts_Crafts_and_Sewing.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Automotive.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Baby.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Beauty.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Cell_Phones_and_Accessories.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Clothing_Shoes_and_Jewelry.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Electronics.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Grocery_and_Gourmet_Food.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Health_and_Personal_Care.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Home_and_Kitchen.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Industrial_and_Scientific.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Musical_Instruments.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Office_Products.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Patio_Lawn_and_Garden.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Pet_Supplies.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Software.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Sports_and_Outdoors.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Tools_and_Home_Improvement.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Toys_and_Games.json.gz
response:200 for =>http://jmcauley.ucsd.edu/data/amazon/qa/qa_Video_Games.json.gz
Total files to process:21
```

The combined file has 1396896 records and 7 columns. Below is a snapshot of the data frame.



	questionType	asin	answerTime	unixTime	question	answerType	answer
0	yes/no	B00000K4MC	Nov 4, 2014	1.415088e+09	Will it work with Windows 8?	Y	You should look here: http://steamcommunity.co...
1	yes/no	B00000K4MC	Nov 29, 2014	1.417248e+09	Will it work on Windows 7 ?	Y	Yes
2	yes/no	B00000K4MC	Aug 16, 2014	1.408172e+09	Is there a digital download of this game?	?	You can buy rollercoaster tycoon Deluxe which ...
3	yes/no	B00000K4MC	Mar 5, 2015	1.425542e+09	will this work on a mac	?	Sorry to tell you Terresa it won't play
4	yes/no	B00000K4MC	Dec 24, 2014	1.419408e+09	I have a new computer with the latest greatest...	Y	this game plays on my lap top just fine ...if ...

FIGURE 1. AMAZON QA RAW DATASET

Amazon QA Dataset Cleaning & Feature Engineering

The useful features to solve the project problem are the fields "question" and "answer". We only considered Questions with maximum length of 50 words or less and Response of maximum length of 10 words or less.

- Text Normalization - Infrequent words and entities like personal names, URLs, email addresses, phone numbers etc. are replaced by special tokens.
- Quotations, punctuations and emojis are removed.
- Salutations like Hi, Hello and Best Regards are removed.
- The language of the message is identified and non-English messages are discarded accordingly.

Twitter Chat Support Dataset

We acquired the twitter dataset from Kaggle which was created by users Thought Vector and Axel StuartBrooke[ii]. Twitter dataset consists of more than 300000 conversation from 2014 to 2017 between twitter support accounts for companies/brands and customers. This data had many appropriate conversations that would be apt for our model and use case.

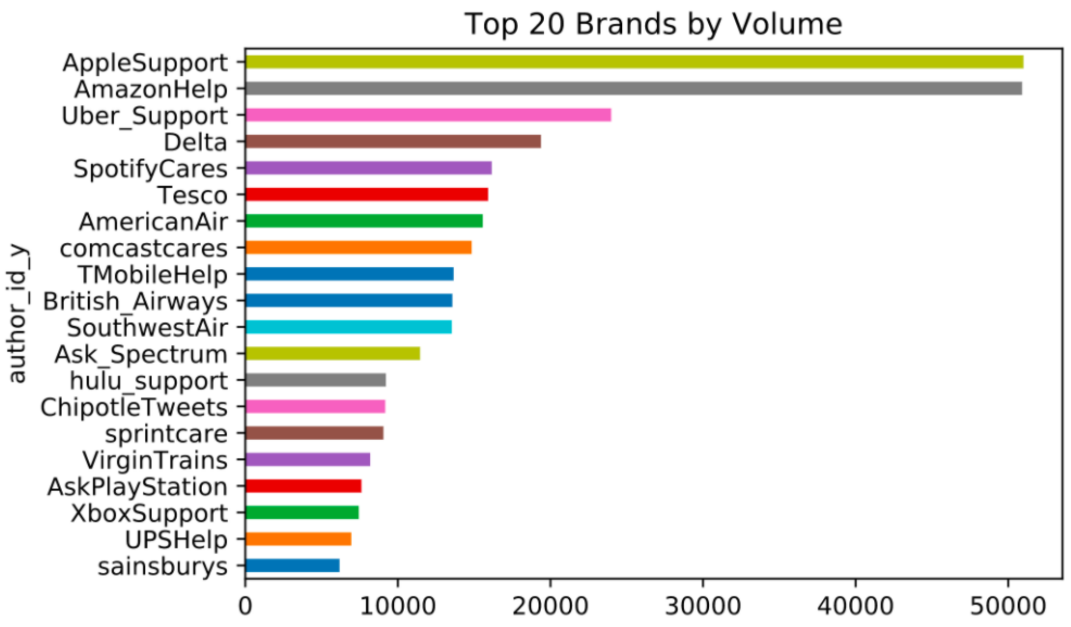


FIGURE 2. NUMBER OF TWEETS FROM BRANDS IN THE DATASET

The dataset is a CSV, where each row is a tweet. The different columns are described below. Every conversation included has at least one request from a consumer and at least one response from a company. Which user IDs are company user IDs can be calculated using the inbound field.

tweet_id A unique, anonymized ID for the Tweet. Referenced by response_tweet_id and in_response_to_tweet_id.

author_id A unique, anonymized user ID. @s in the dataset have been replaced with their associated anonymized user ID.

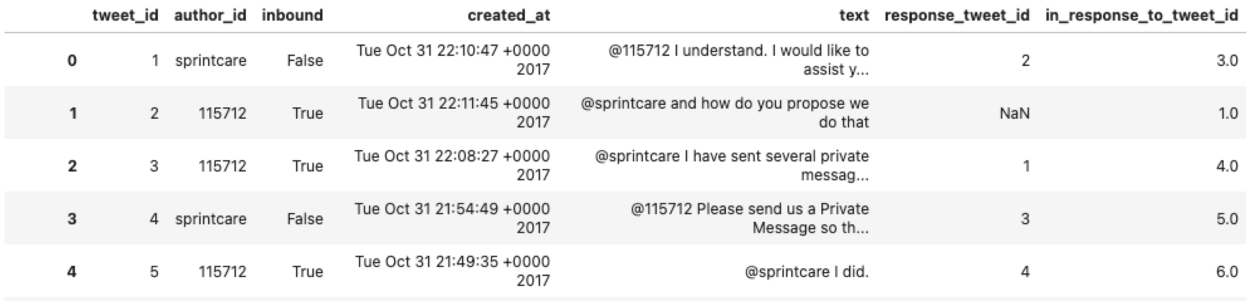
Inbound Whether the tweet is "inbound" to a company doing customer support on Twitter. This feature is useful when re-organizing data for training conversational models.

created_at Date and time when the tweet were sent.

Text Tweet content. Sensitive information like phone numbers and email addresses are replaced with mask values like __email__.

response_tweet_id IDs of tweets that are responses to this tweet, comma separated.

in_response_to_tweet_id ID of the tweet this tweet is in response to, if any.



	tweet_id	author_id	inbound	created_at	text	response_tweet_id	in_response_to_tweet_id
0	1	sprintcare	False	Tue Oct 31 22:10:47 +0000 2017	@115712 I understand. I would like to assist y...	2	3.0
1	2	115712	True	Tue Oct 31 22:11:45 +0000 2017	@sprintcare and how do you propose we do that	NaN	1.0
2	3	115712	True	Tue Oct 31 22:08:27 +0000 2017	@sprintcare I have sent several private messag...	1	4.0
3	4	sprintcare	False	Tue Oct 31 21:54:49 +0000 2017	@115712 Please send us a Private Message so th...	3	5.0
4	5	115712	True	Tue Oct 31 21:49:35 +0000 2017	@sprintcare I did.	4	6.0

FIGURE 3. TWITTER RAW DATASET

Twitter Dataset Cleaning

We wanted only sentences and words with intent and meaningful responses. So, we cleaned the data for following aspects:

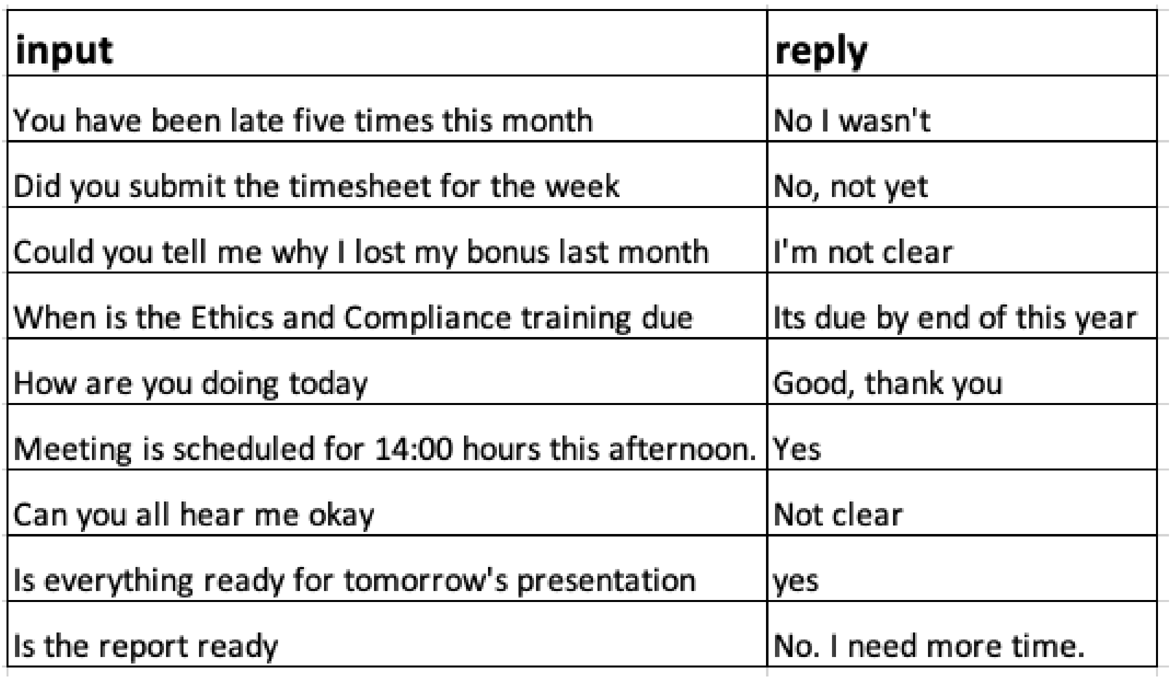
1. Cleaned all the hashtags (for example: '#helpneeded) and all the mentions of user Ids (for example: '@AppleSupport).
2. Cleaned Emojis
3. Removed URLs mentioned in the text (for example: https://www.google.com)
4. Removed punctuation from the sentences.
5. Changed the text to lower case.
6. Removed single letter texts (for example: 'k')
7. Removed other language texts.
8. Removed Null and duplicate rows.

Twitter Dataset Feature Engineering

We created our features 'input' and 'reply' which are uniform across our other datasets. We've used "in_reposne_to_tweet_id", "response_tweet_id" and "tweet_id" columns to recognize messages and corresponding replies. We've limited our input text size to more than one word and less than 50 words. We've limited our reply text size to more than one word and less than 10 words.
We've created a new twitter dataset with 'input' and 'reply' columns with 268992 records.

Custom Dataset

We've created a custom dataset with 500 records to add instances that'd improve and attune the model for the business use case. Similar to our other datasets we've created data with input and reply columns with the similar restrictions of 50 words and less for input messages and 10 words and less for reply messages.



input	reply
You have been late five times this month	No I wasn't
Did you submit the timesheet for the week	No, not yet
Could you tell me why I lost my bonus last month	I'm not clear
When is the Ethics and Compliance training due	Its due by end of this year
How are you doing today	Good, thank you
Meeting is scheduled for 14:00 hours this afternoon.	Yes
Can you all hear me okay	Not clear
Is everything ready for tomorrow's presentation	yes
Is the report ready	No. I need more time.

FIGURE 4. SNAPSHOT OF OUR CUSTOM DATASET

Dataset	No. of records
Amazon Q/A Dataset	25000
Twitter Support Chat	268992
Custom Dataset	500

Descriptive Analytics

To perform Exploratory Data Analysis on the Amazon QA dataset, we focused mainly on the two features ‘Questions’ and ‘Answers’. The below figure gives us a plot of open-ended questions compared to yes/no questions and their corresponding counts. From this analysis, we can conclude that final combined dataset contains a higher number of open-ended questions than simple yes/no questions.

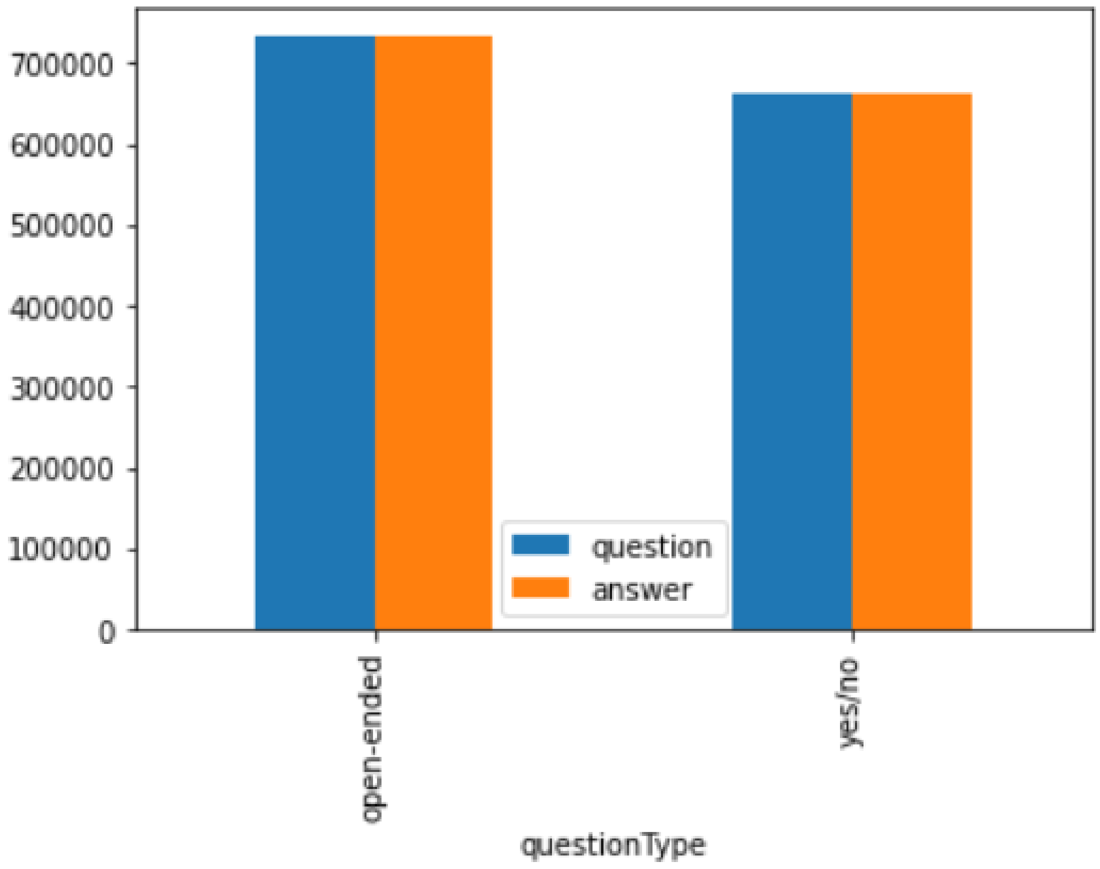
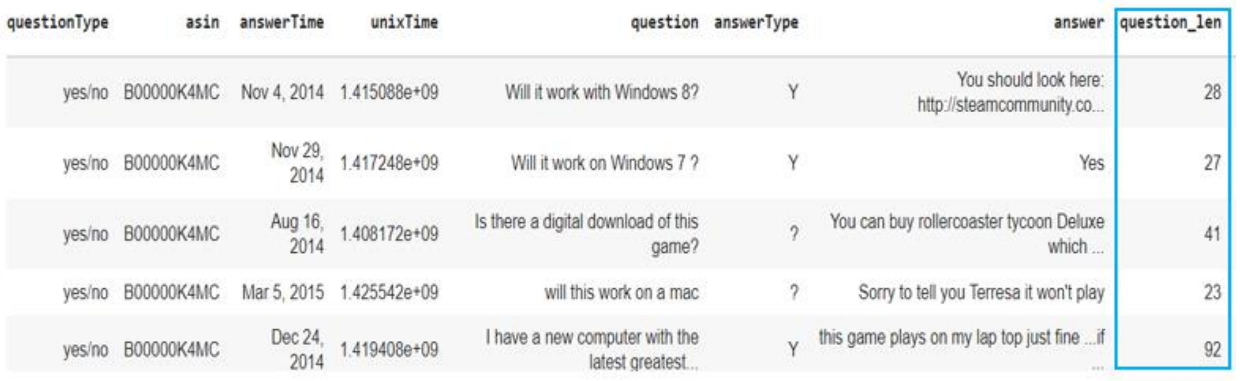


FIGURE 5. TYPE OF QUESTION

We calculated the length of each question found in the combined dataset. Based on the data analysis, as provided below, it is safe to assume that most of the questions have more than 20 words. We observed that the length of most of the questions in the Amazon dataset is around **50 words**.



questionType	asin	answerTime	unixTime	question	answerType	answer	question_len
yes/no	B00000K4MC	Nov 4, 2014	1.415088e+09	Will it work with Windows 8?	Y	You should look here: http://steamcommunity.co...	28
yes/no	B00000K4MC	Nov 29, 2014	1.417248e+09	Will it work on Windows 7 ?	Y	Yes	27
yes/no	B00000K4MC	Aug 16, 2014	1.408172e+09	Is there a digital download of this game?	?	You can buy rollercoaster tycoon Deluxe which ...	41
yes/no	B00000K4MC	Mar 5, 2015	1.425542e+09	will this work on a mac	?	Sorry to tell you Terresa it won't play	23
yes/no	B00000K4MC	Dec 24, 2014	1.419408e+09	I have a new computer with the latest greatest...	Y	this game plays on my lap top just fine ...if ...	92

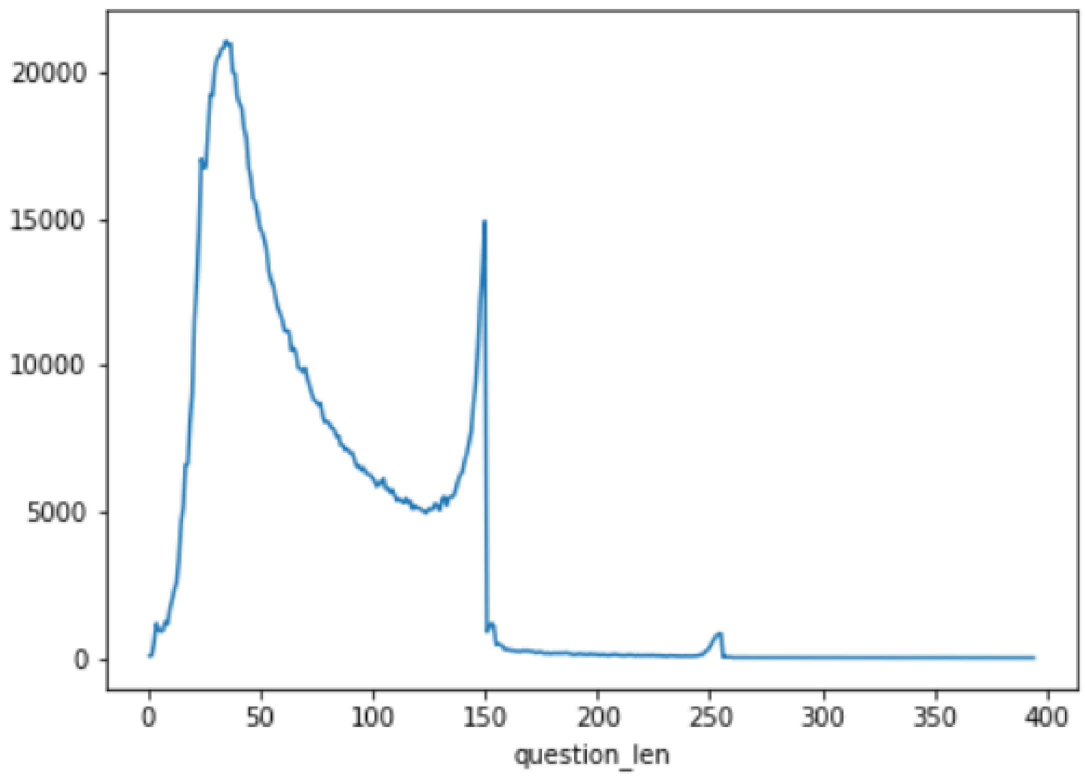


Figure 6. Length of Question

Based on our data analysis, we can conclude that the dataset contains a wide variety of questions with common first two words like ‘does this’, ‘Is this’, ‘Will this’.

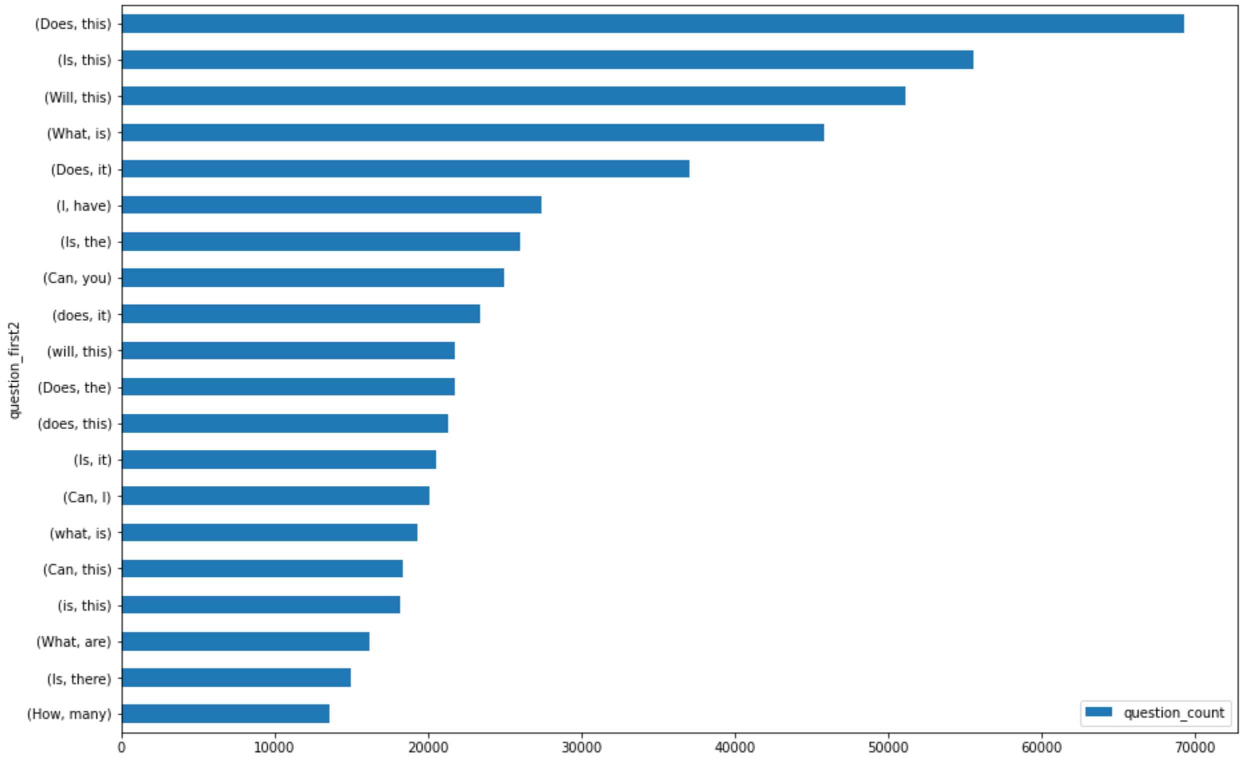


FIGURE 7. FIRST 2 WORDS OF QUESTIONS

We have also observed that there are a lot of responses with the same first 2 words. Per the visual below, the count of such answers is around 35000.

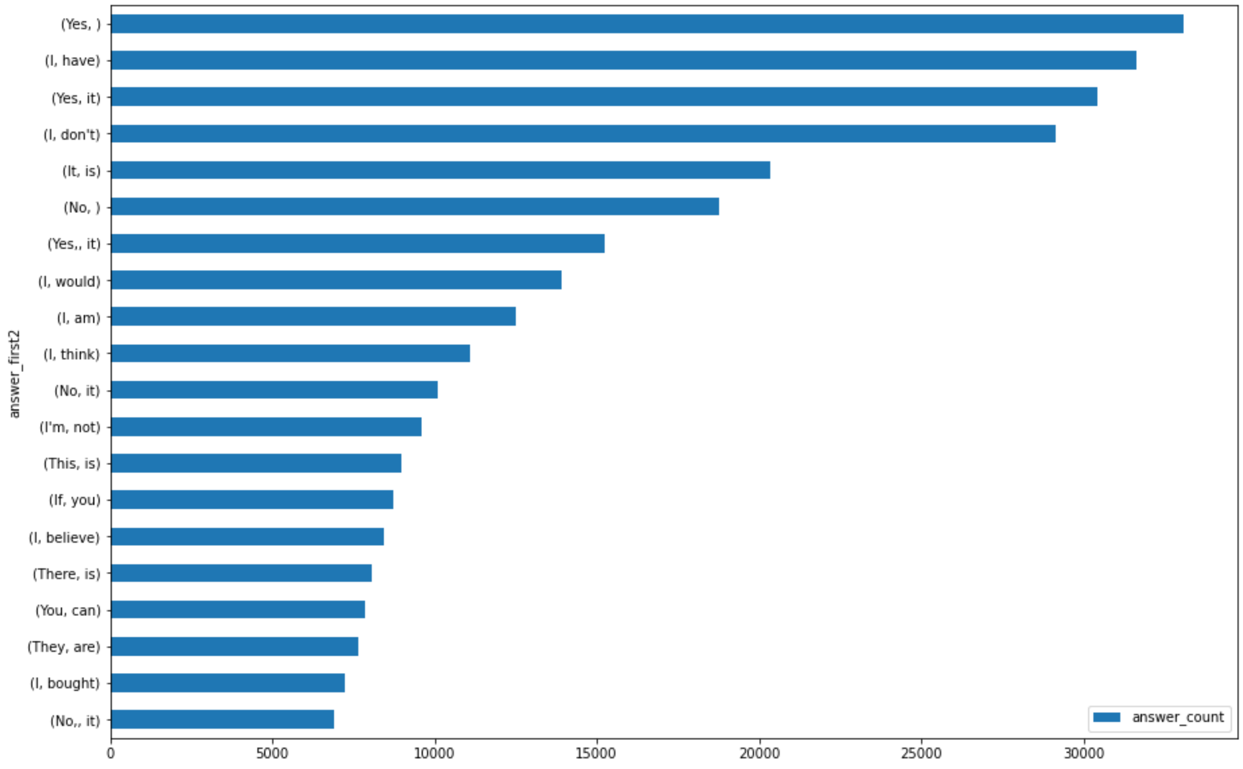


FIGURE 8. FIRST 2 WORDS OF RESPONSES

As per our EDA, we believe Amazon QA dataset would be appropriate for our deep learning modeling in order to suggest the best responses.

Project Methods

Our Approach is based on the Google's smart reply paper[iii] and on this article from towards data science blog[iv]. uses three methods in the process. As the process is divide into two steps, the first step being creating reply types or response sets by clustering the replies. We've used Sent2vec technique to vectorize the replies and then Density-Based Spatial Clustering of Application with Noise (or commonly known as DBSACN Algorithm) to create the reply types. We cluster or group the replies based on their intent or semantic similarity between replies.

Reply Type Creation

We've used sent2vec vectorization technique because it is easily implemented but also provides with best results. Sent2vec uses Pre trained BERT or Bidirectionally Encoder Representations from Transformers[v], a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. Using Pretrained BERT model Sent2vec converts sentences (replies here) to vectors which are points in N-dimensional space. Sent2vec provides default vectors of size 768(N=768). So all the semantically similar sentences are closer in the N dimensional space.

After we use DBSCAN Clustering algorithm, we chose this algorithm for multiple reasons:

1. The algorithm uses Cosine Similarity as metric to cluster the datapoints.
2. We do not need to provide initial hyperparameters like number of clusters like in k-means clustering where you have to provide k. Since we do not know how many clusters will form from the data, this is the best option.
3. It identifies noise points – Noise points are data points which didn't belong in any cluster, so in this context they become random or very context specific responses that wouldn't suit our approach.

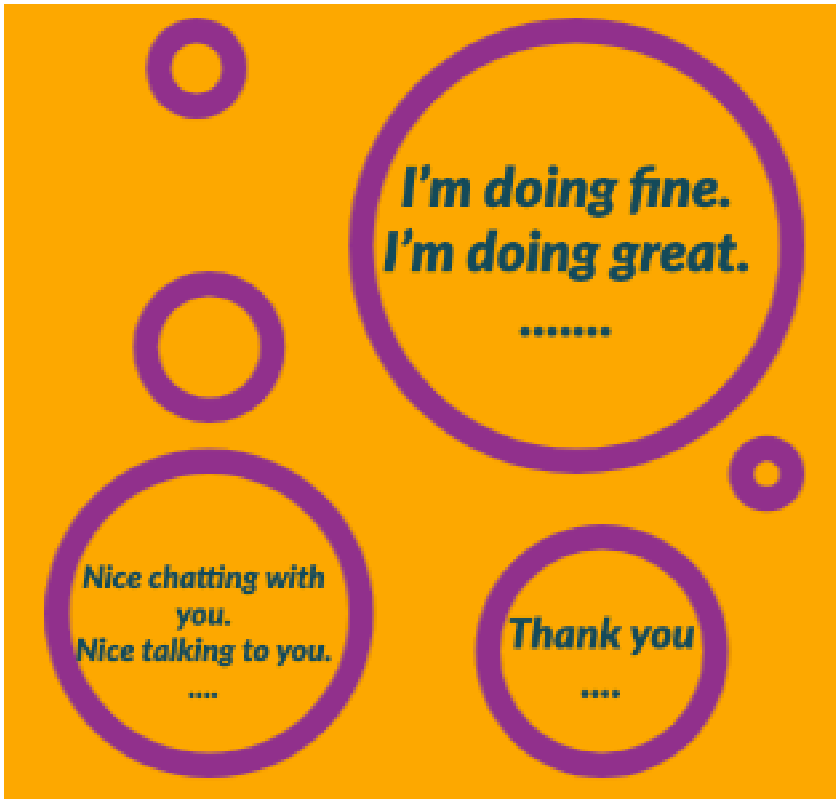


FIGURE 9. REPLY TYPE CLUSTERS VISUALIZATION

Reply Prediction for Incoming Message

We take input messages to train the LSTM model or Long Short Term Memory neural network model to predict the top three reply types that are clustered by DBSCAN. By doing this instead of predicting one message using Seq2Seq methods, we can provide multiple predictions that aren't of same intent type. We can provide three predictions or more of different intent. For example: for an incoming message "Can you Join the zoom call?", we can suggest three replies with different intents like "Yes, I can", "No, I can't", "Not sure". Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. LSTM model is used instead other types of models because, by using this we can predict probabilities of all reply type clusters and choose the top 3 or however many we need. But the main reason to choose an LSTM is that this model, reads the sentence and takes the context/dependencies from each word in the sentence uses it to provide output.

The incoming messages gets tokenized and are fed to the model which contains an Embedding layer, an LSTM layer and final Fully connected hidden layer to produce probabilities of all the reply type clusters for a corresponding message.

LSTM takes in Incoming messages and its output will be labels of reply clusters which are generated from DBSCAN clustering.

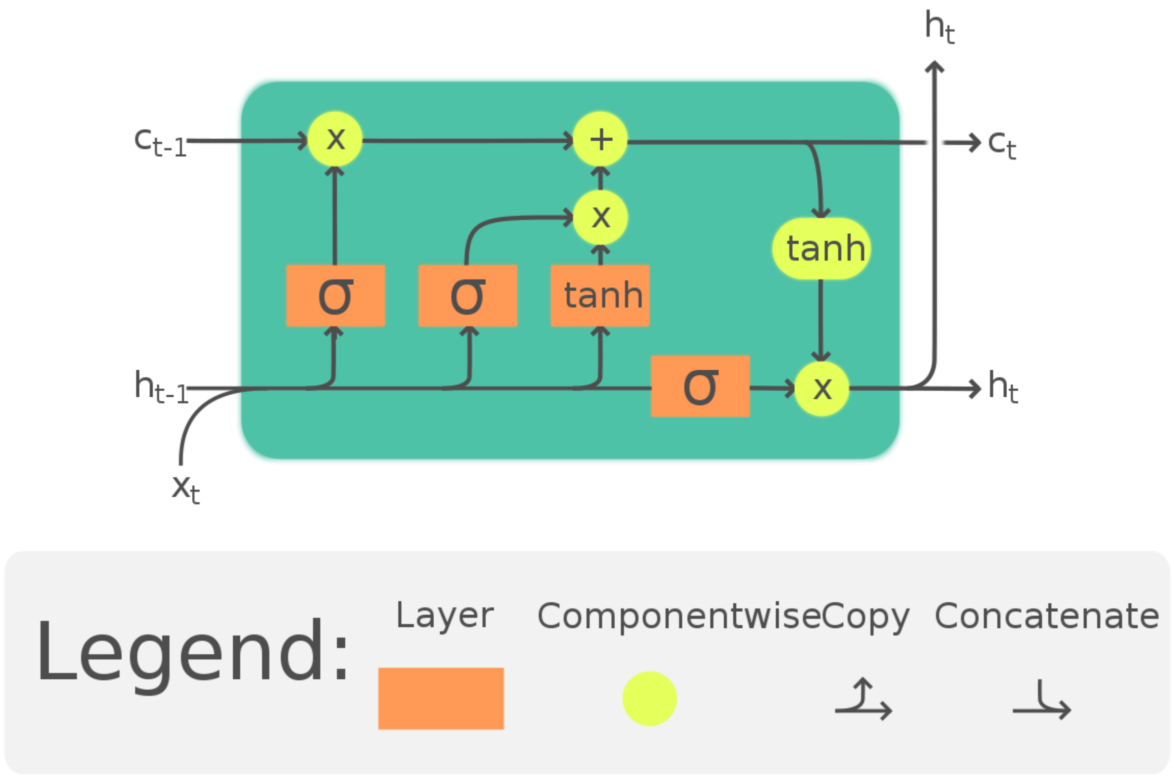


FIGURE 10. LSTM ARCHITECTURE

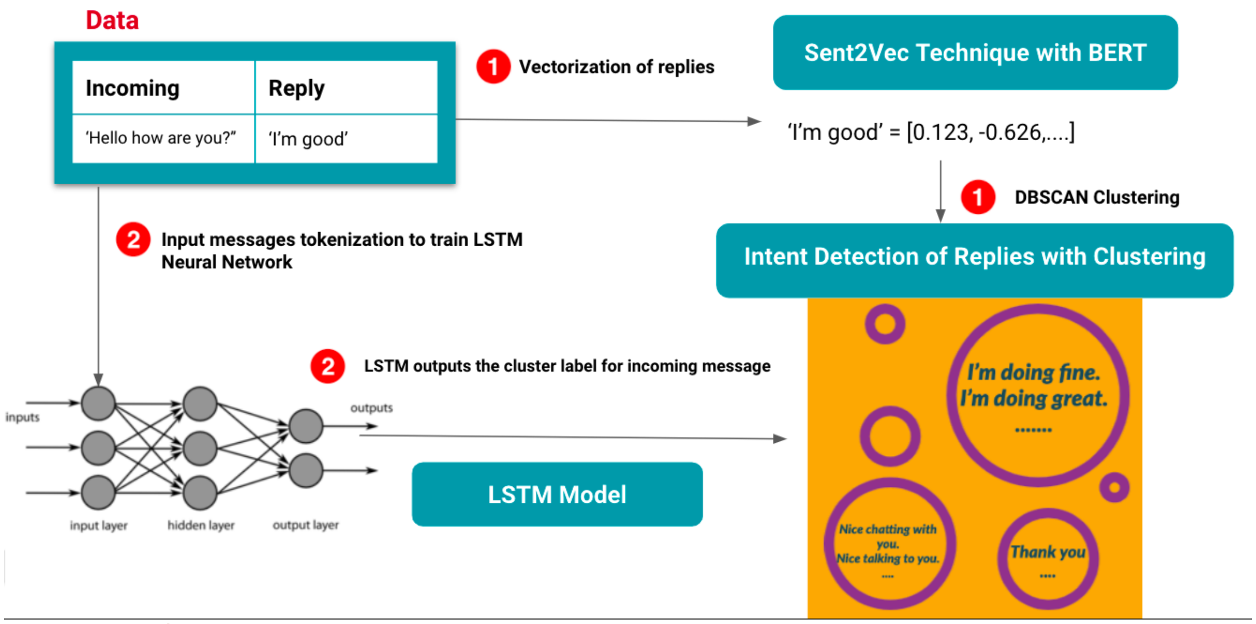

Data
Incoming
Reply
'Hello how are you?"
'I'm good'
1 Vectorization of replies
Sent2Vec Technique with BERT
'I'm good' = [0.123, -0.626,....]
1 DBSCAN Clustering
Intent Detection of Replies with Clustering
2 Input messages tokenization to train LSTM Neural Network
2 LSTM outputs the cluster label for incoming message
inputs
outputs
input layer
hidden layer
output layer
LSTM Model
I'm doing fine.
I'm doing great.
Nice chatting with you.
Nice talking to you.
Thank you

FIGURE 11. OVERVIEW OF OUR MODEL

All the data cleaning, data preparation and data wrangling was done using pandas on python. Modeling Building was for LSTM model was done on PyTorch, and Clustering model on LSTM. Other Python Packages like NumPy, transformers, genism, Sent2vec were used for other data preparation and Vectorization process.

Results and Interpretations

We started trying out our individual datasets with the model. We first used our twitter dataset and vectorized the replies. Then we used the DBSCAN algorithm on the replies.
The Hyperparameters of this algorithm are Epsilon, Distance metric used, minimum number of samples per cluster.

Epsilon: Two points are considered neighbors/or part of same cluster if the distance between the two points is below the threshold epsilon.

Minimum number of samples per cluster: The minimum number of neighbors a given point should have in order to be classified as a core point.

Distance Metric: The metric to use when calculating distance between instances in a feature array (i.e., Euclidean distance, Cosine similarity).

Our results from clustering the twitter data replies:

- Number of clusters: 2470

- Number of noise points: 222650
- Total records: 268992

The Hyper parameter used in the algorithm are:

- Epsilon = 0.0012
- Minimum number of samples per cluster = 2
- Metric = cosine similarity

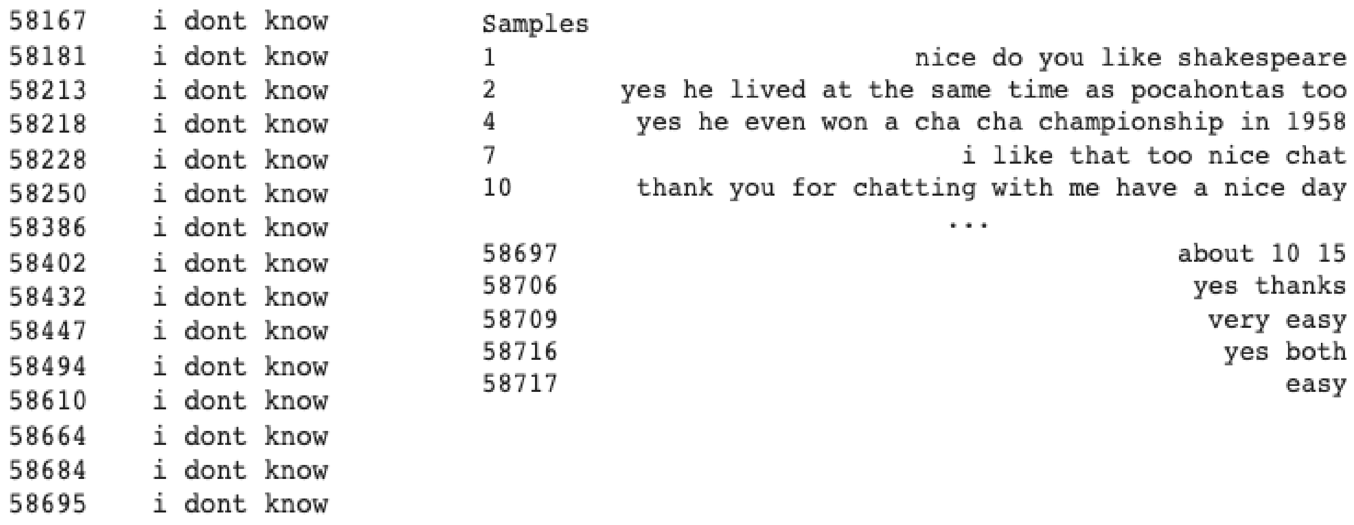
Bias in the model and Noise Point Elimination

We used these hyperparameters to cluster the sentences very tightly to put all the similar sentences in same cluster even if there are two replies that are similar. As you can see there are 222650 Noise points detected by the model. All these noise points are placed into cluster labelled '-1' by the model. We have used the LSTM model on these results and the model gave us an accuracy of 81%.

Hyper Parameters used:

- Embedding Dimension =128
- Hidden layer size =128
- Learning rate = 0.001
- Optimizer = Adam

But the noise points are 222650 out of 268992. We got to know that most of the predictions are for the noise cluster, so we got a false accuracy. The -1 cluster is causing bias in the LSTM model, we confirmed this again by removing the noise points from the data and again performed the clustering and LSTM model and got a validation accuracy of 41%.



```
58167    i dont know
58181    i dont know
58213    i dont know
58218    i dont know
58228    i dont know
58250    i dont know
58386    i dont know
58402    i dont know
58432    i dont know
58447    i dont know
58494    i dont know
58610    i dont know
58664    i dont know
58684    i dont know
58695    i dont know
```

```
Samples
1                                   nice do you like shakespeare
2          yes he lived at the same time as pocahontas too
4           yes he even won a cha cha championship in 1958
7                                      i like that too nice chat
10          thank you for chatting with me have a nice day
                                  ...
58697                                               about 10 15
58706                                                yes thanks
58709                                                 very easy
58716                                                  yes both
58717                                                      easy
```

FIGURE 12. REGULAR CLUSTER WITH SIMILAR SENTENCES ON THE LEFT AND NOISE CLUSTER ON THE RIGHT

Even for the purpose of the final results and use case of the model, these types of replies are not desirable. So, we considered the removal of noise points from datasets as final solution for this problem. So, we performed the clustering on all externally sourced datasets and removed noise data points from each dataset. We combined all the remaining records in all datasets and created a new final dataset with 58718 records. Then we divided the data into Training -Validation-Test data with ratio of 70-20-10 and used the training dataset to train the entire models with same previous hyperparameters. We got an improved result of 46.19%. So, combining data yielded better results and is helping in creating reply types that are much varied. We put the whole dataset through clustering to create reply types and create a new 'label' column with reply type cluster label to which the reply belongs.

Hyperparameter Tuning for DBSCAN Clustering and LSTM

Since the bias and noise from models is eliminated. We need to tune hyperparameters for both models to improve model performance.

DBSCAN Model

For DBSACN model, we increased number of minimum samples per clusters to 4 to create robust reply types and generalize the model, so reply types with at least 4 types of replies form a cluster.
Coming to epsilon, which is our main hyperparameter for DBSCAN model, we tried our multiple values of epsilon and plotted our results to find the relation between epsilon and how number of clusters and number of noise points are varying. Also, we checked cosine similarity, which we discussed is a metric to measure sentence similarity between any two given sentences. So, we plotted average cosine similarity of all clusters across various epsilon.

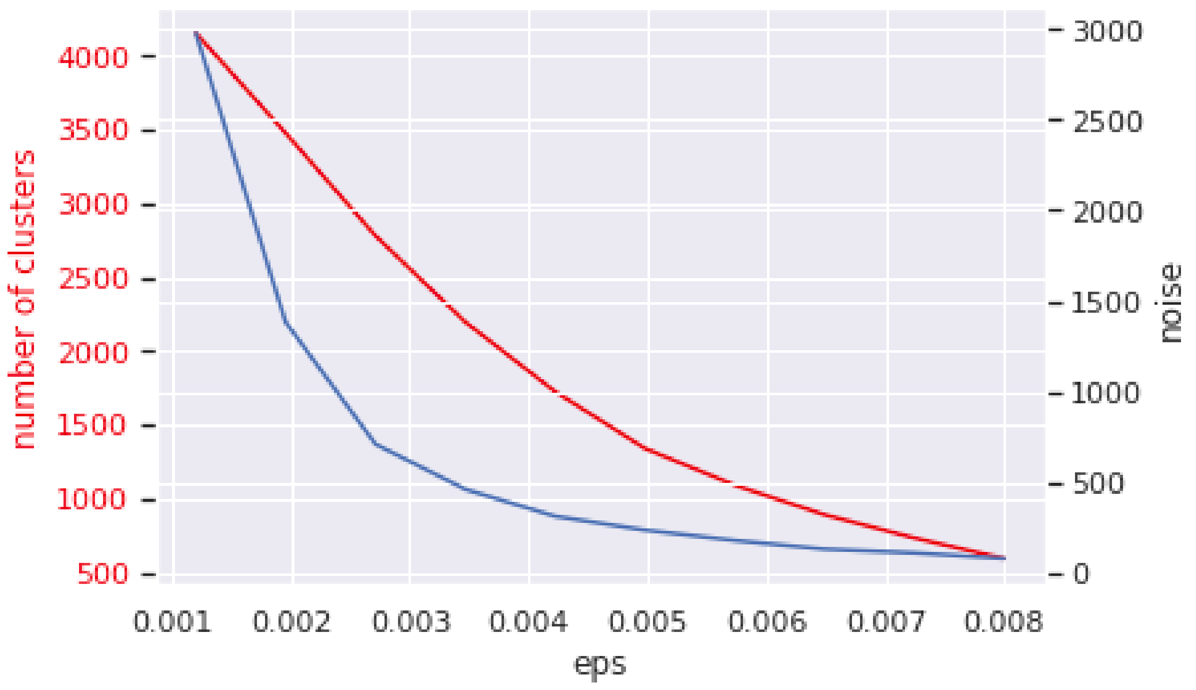


FIGURE 13. NUMBER OF CLUSTERS AND NOISE VS EPSILON

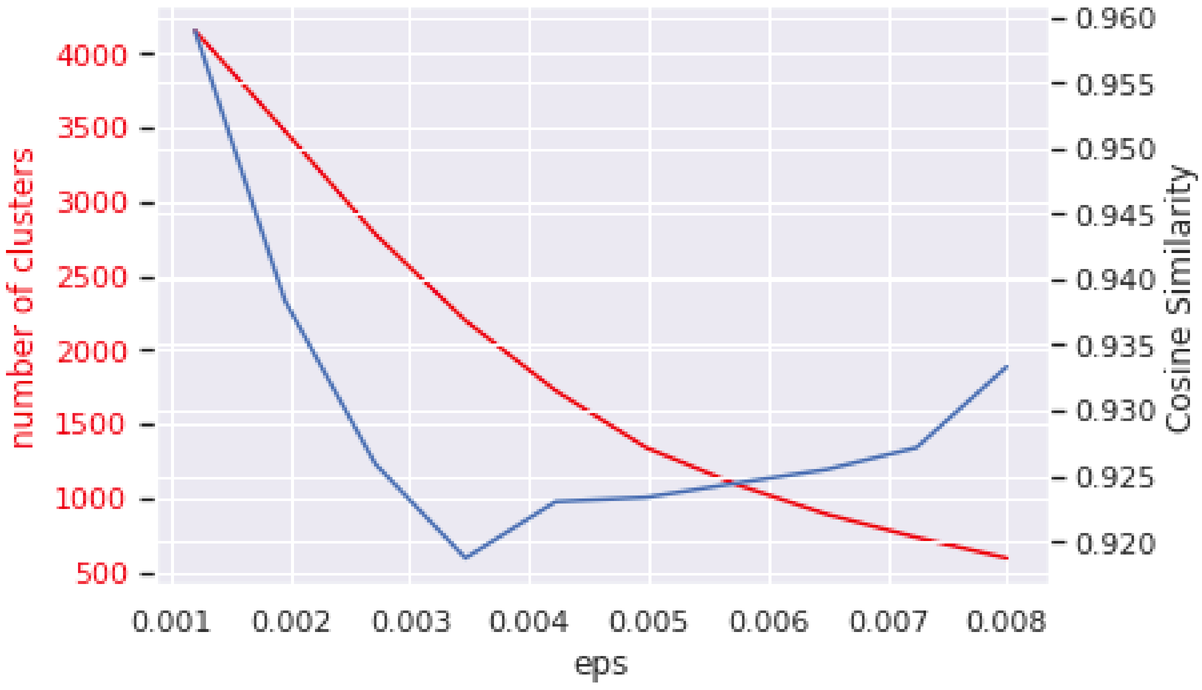


FIGURE 14. AVERAGE COSINE SIMILARITY ACROSS CLUSTERS VS EPSILON

You can observe that number of clusters decrease from over 4000 to around 600 for epsilon varying from 0.001 to 0.008. The clusters are decreasing as we increase epsilon and at the same time you see that average cosine similarity in lusters is highest at lower epsilon values and decreases as we increase epsilon. This indicates using lower epsilon we get tighter clusters with replies or sentences which are very similar, but as we keep increasing the epsilon or distance between points to be considered neighbor the clusters get looser with sentences that are not exactly similar. Lower epsilon basically overfits the clusters and higher epsilon values under fit clusters. Sentence similarity should keep decreasing as we increase epsilon, but we see a rise in cosine similarity again, which was confusing at first but we think it was due to twitter dataset which contains very similar types of replies as customer service agents use very similar and uniform language. Also, High number of clusters cause efficiency problems for the LSTM model down the line, LSTM may have tough job to pick up 3 reply types from 4000 reply types than picking three from 1000 reply types. The multi-label classification problem will also be solved by generalizing the clustering model. So, to generalize our model we chose an epsilon value of 0.0044.

Final DBSCAN clustering model results:

- Taking Epsilon = 0.0044
- Minimum samples in cluster = 4
- Metric = cosine similarity

- Estimated number of clusters: 1016
- Estimated number of noise points: 1696

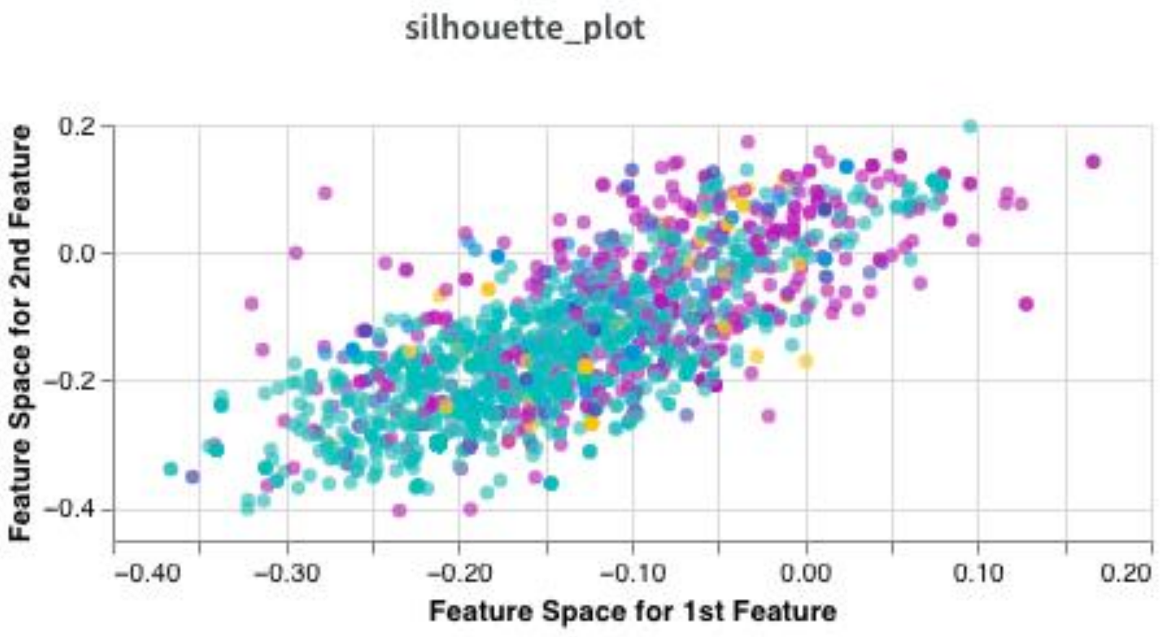


FIGURE 15. CLUSTERS VISUALIZED IN THE FEATURE SPACE

LSTM Model

The hyperparameters in our LSTM model architecture are:

Embedding dimension – The dimension of the vectors the embedding layer will convert the input tokens. We've used 128, 256, 512 as these are regularly used values for the embedding dimensions and in general for hidden layers in neural networks.

Hidden size – Size of the hidden layer for LSTM model. We've also used the values 128, 256, 512 here.

Optimizer - Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate to reduce the losses[vi]. We tried both SGD (Stochastic Gradient Descent and Adaptive Moment Estimation) for the LSTM model.

Learning rate - A hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. The values tested were from 0.1 to 0.0001.

Epochs – The number of times the model should run forward and backward to train. We tried the values from 15 to 25.

We performed hyperparameter tuning using sweeps from weights and Biases using training dataset and checking the validation accuracy on validation dataset. Sweeps is tool that takes all the different hyperparameters in the model and tries their combinations for the goal of either maximizing the Accuracy or Decreasing the loss or other goals. We tested our hyperparameters by random search.

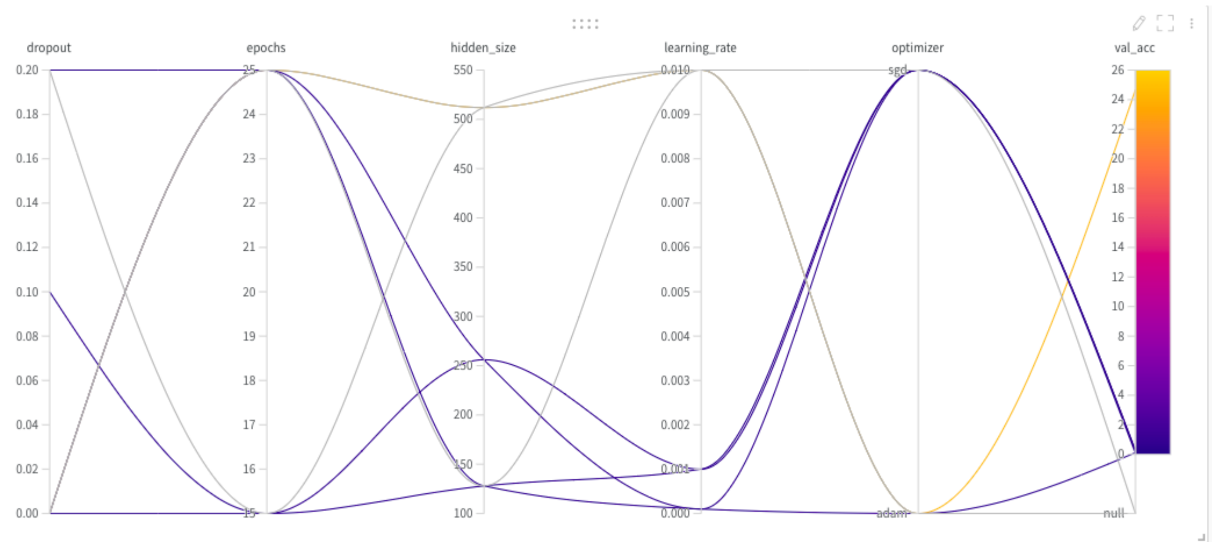


FIGURE 16. SWEEPS TESTING FOR DIFFERENT HYPERPARAMETERS

We have found our best combination of hyperparameters and used them to train our final model:

Embedding dimension = 128
Hidden size = 128
Learning rate = 0.001
Optimizer = Adam
Epochs = 15

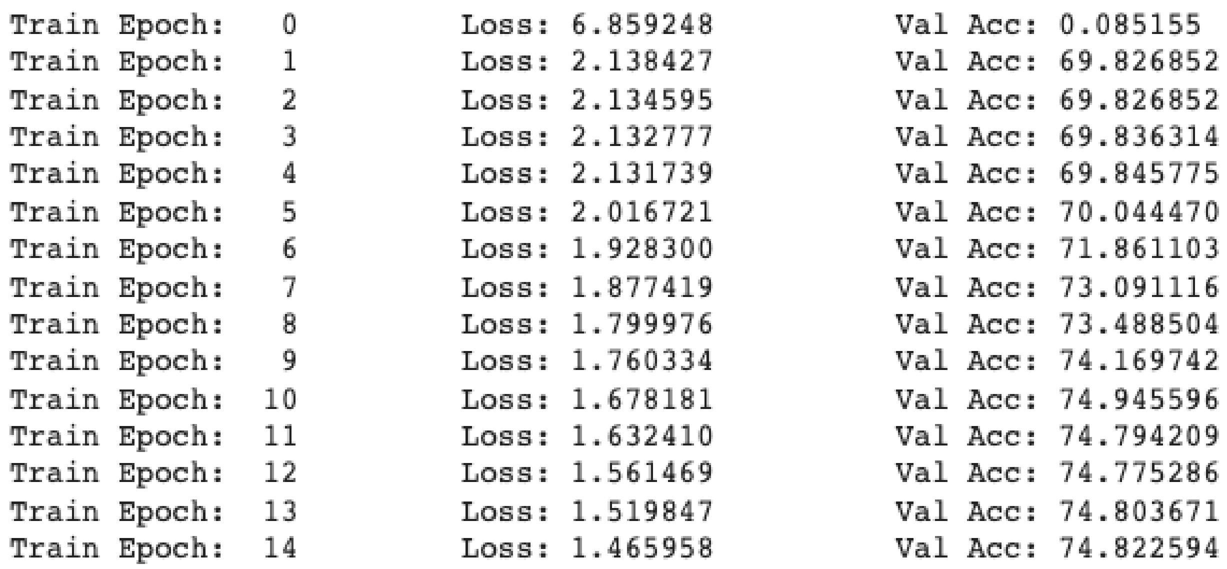
We've used three metrics to measure LSTM model Performance:

1. Cross-Entropy Loss - Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events. It is measured between output from the LSTM model probabilities and Ground truths (Actual Labels in the validation dataset.

2. Validation Accuracy – If one of the three predictions is correct as to what is there as ground truth labels.

3. Cosine Similarity – Like previously discussed, it is a metric to measure sentence similarity from 0 to 1. We take average cosine similarity of our predictions with the ground truth.

For cross-entropy loss we achieved 1.465958. Here's the trajectory of Loss function during model training.



```
Train Epoch:    0          Loss: 6.859248          Val Acc: 0.085155
Train Epoch:    1          Loss: 2.138427          Val Acc: 69.826852
Train Epoch:    2          Loss: 2.134595          Val Acc: 69.826852
Train Epoch:    3          Loss: 2.132777          Val Acc: 69.836314
Train Epoch:    4          Loss: 2.131739          Val Acc: 69.845775
Train Epoch:    5          Loss: 2.016721          Val Acc: 70.044470
Train Epoch:    6          Loss: 1.928300          Val Acc: 71.861103
Train Epoch:    7          Loss: 1.877419          Val Acc: 73.091116
Train Epoch:    8          Loss: 1.799976          Val Acc: 73.488504
Train Epoch:    9          Loss: 1.760334          Val Acc: 74.169742
Train Epoch:   10          Loss: 1.678181          Val Acc: 74.945596
Train Epoch:   11          Loss: 1.632410          Val Acc: 74.794209
Train Epoch:   12          Loss: 1.561469          Val Acc: 74.775286
Train Epoch:   13          Loss: 1.519847          Val Acc: 74.803671
Train Epoch:   14          Loss: 1.465958          Val Acc: 74.822594
```

FIGURE17. TRAINING OF MODEL AND CHANGE OF LOSS AND ACCURACY ACROSS EPOCHS

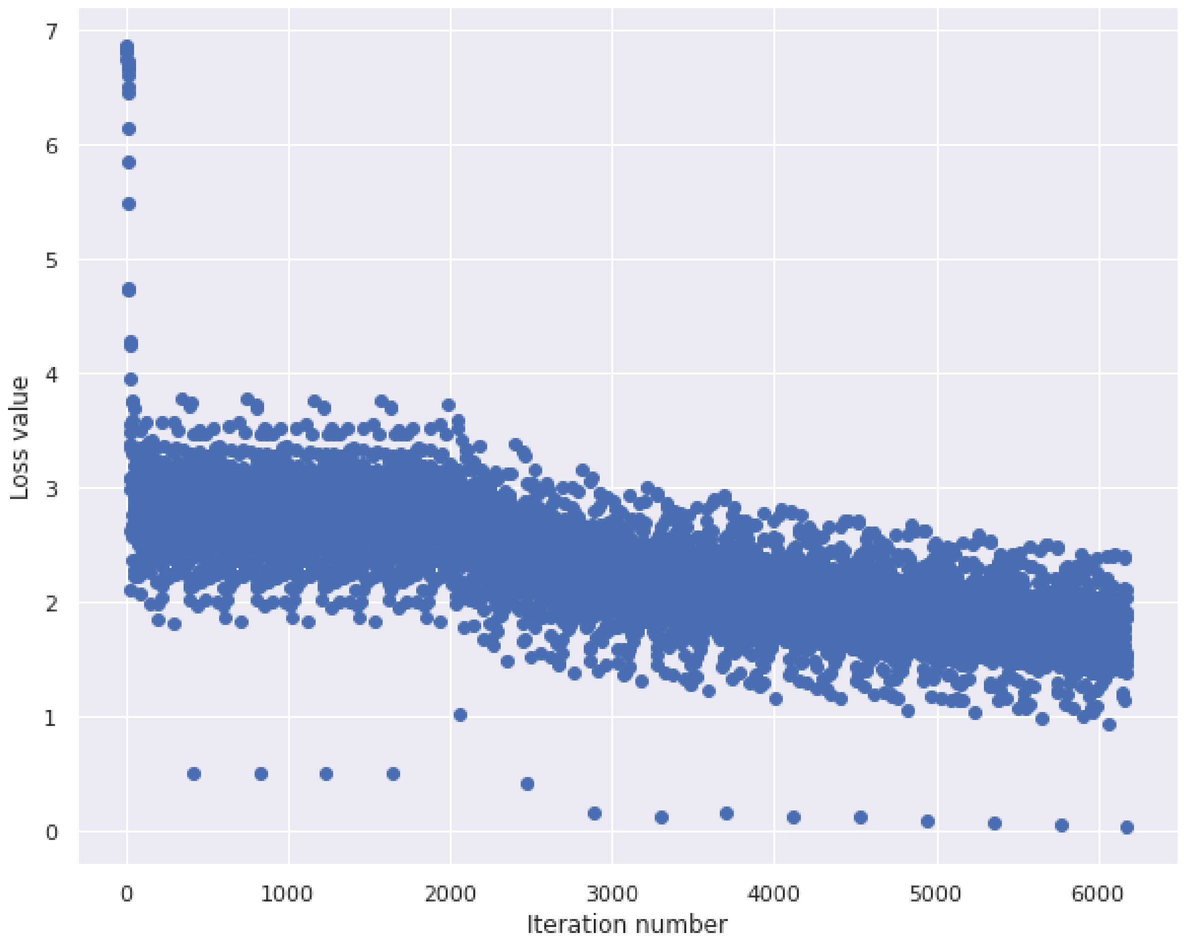


FIGURE18. TRAJECTORY OF LOSS ACROSS TRAINING

Validation Accuracy of our model is 74.82%, which is pretty good as the model is predicting correct reponse type from the 1016 reply clusters.

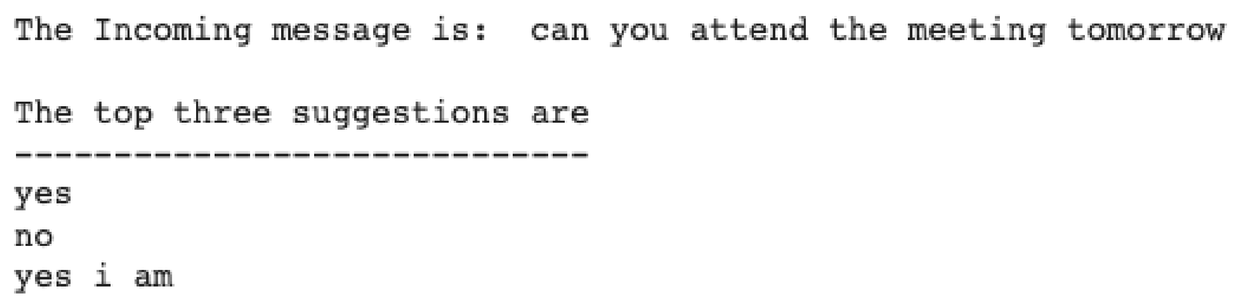
The average cosine similarity for our predictions is 0.368. Even though it isn't high, we do not seek cosine similarity as our primary metric because we do not want exact replies from our model but more short and generalized replies that work regardless of context of messages.

So, our model's primary metric would be Validation accuracy which is metric for guessing correct reply type for incoming messages.

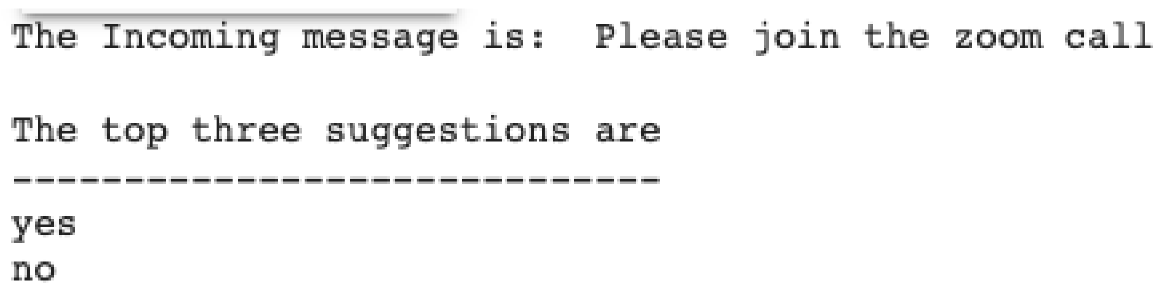
The smaller number of reply type clusters the better LSTM model performance because easier task to do multi label classification on less number of labels. We've also sacrificed tighter and closer reply clusters to generalize the clustering model to form reply clusters with varied replies of sematic similarity or meaning in same cluster.

After that, we need to filter out reply clusters that are irrelevant or inappropriate to our use case manually. We filtered out those clusters and also assigned a default reply to each cluster, so replies would be consistent and short. We can automate this process too by selecting a centroid of each cluster by vectorizing the sentences and finding a centroid of the sentences in cluster. We can also create and train a model to check for irrelevant and inappropriate reply type clusters. But mostly the model needs better cleaned and use case augmented data for training, which would provide better results as the model is very sensitive to the data.

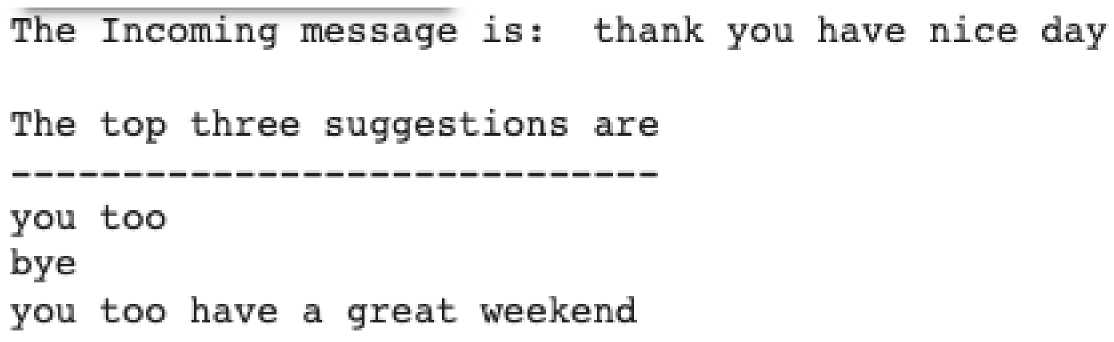
Sample results from our model



```
The Incoming message is:  can you attend the meeting tomorrow

The top three suggestions are
-----------------------------
yes
no
yes i am
```

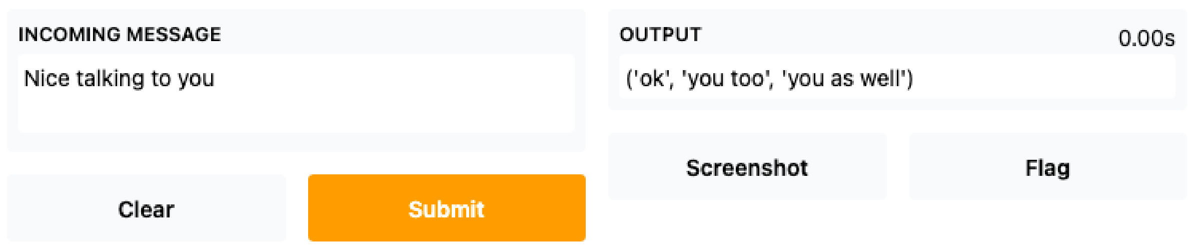


```
The Incoming message is:  Please join the zoom call

The top three suggestions are
-----------------------------
yes
no
```



```
The Incoming message is:  thank you have nice day

The top three suggestions are
-----------------------------
you too
bye
you too have a great weekend
```

We've also implemented Gradio[vii] UI web app to present the company how the final product may look like and how fast it can be.



INCOMING MESSAGE

Nice talking to you

OUTPUT 0.00s

('ok', 'you too', 'you as well')

FIGURE 19. RESULTS ON THE GRADIO WEB APP

Additional Modelling

We tried using HDBSCAN (hierarchical DBSCAN) model instead of DBSCAN but HDBSCAN doesn't use cosine similarity as distance metric to cluster its data points, so it's not helpful in this situation where we cosine similarity is the best metric for sentence similarity.

We tried another model called correlation matrix model, where instead of the LSTM in the previous model we use DBSCAN clustering on vectorized incoming messages using sent2vec creating incoming message type clusters. We created correlation matrix for incoming clusters and reply clusters, so we would know the probability of each reply clusters for each incoming message cluster. This model did not work with 0.73% validation accuracy, and we think because man questions/incoming messages are not at all similar and widely varied, so many of them came across as noise to the algorithm and went to -1 noise cluster. This causes the problem for the model to recognize the appropriate reply cluster types to incoming message types.

Next, we used Support vector Machine on vectorized incoming messages using sent2vec instead of LSTM because it would be perfect to classify vectors, but the problem/defect of the model is that it can only provide one suggestion/prediction.

We also used Multi layered perceptron instead of LSTM on vectorized incoming messages using sent2vec but achieved a validation accuracy of 61%. So, all around LSTM performed better in all aspects than other models.

Recommendations for Future Work

For future work Our recommendations are:

1. Data Collection – data needs to be more robust and much varied and relevant to the model's use case.

2. Model Building – Using Attention based Transformer model instead of LSTM because this model takes more information from sentences like local context of words and can provide better results.

3. Trigger models to check whether the incoming models needs suggestions.

4. Model to weed out irrelevant and inappropriate reply cluster.

5. Automatically assigning a default reply to each cluster by finding its centroid or something similar in sense.

6. Using Sent2vec vectorizer instead of embedding layer for LSTM model.

Conclusion

Our model works better than available seq2seq models that predict replies for incoming messages because our models can suggest replies with different intents and our model was successful with most of the use cases but It can be better when provided with more augmented data. DBSCAN clustering works perfectly for clustering vectorized replies. LSTM is providing 74.82% validation accuracy for predicting correct reply types. This model can be put in production by training it with more chat data collected from professional users for usage on the platform.

---

[i] https://towardsdatascience.com/training-your-own-message-suggestions-model-using-deep-learning-3609c0057ba8
[ii] https://www.kaggle.com/soaxelbrooke
[iii] https://arxiv.org/pdf/1606.04870.pdf
[iv] https://towardsdatascience.com/training-your-own-message-suggestions-model-using-deep-learning-3609c0057ba8
[v] https://arxiv.org/pdf/1810.04805.pdf
[vi] https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6
[vii] https://gradio.app/